BLACK CODHER

CODING PROGRAMME

CODING BLACK FEMALES

Black Codher Bootcamp

NÍ YỌ

BLACK CODHER

CODING PROGRAMME

UNIT 3
jQuery

RECAP

- Modifying the **DOM** with JavaScript

WHAT YOU'LL BE LEARNING TODAY?

- How to use a JavaScript library
- The jQuery `$()` function

LEARNING OUTCOMES

Building a TODO list with jQuery 🎉

BLACK CODHER

CODING PROGRAMME

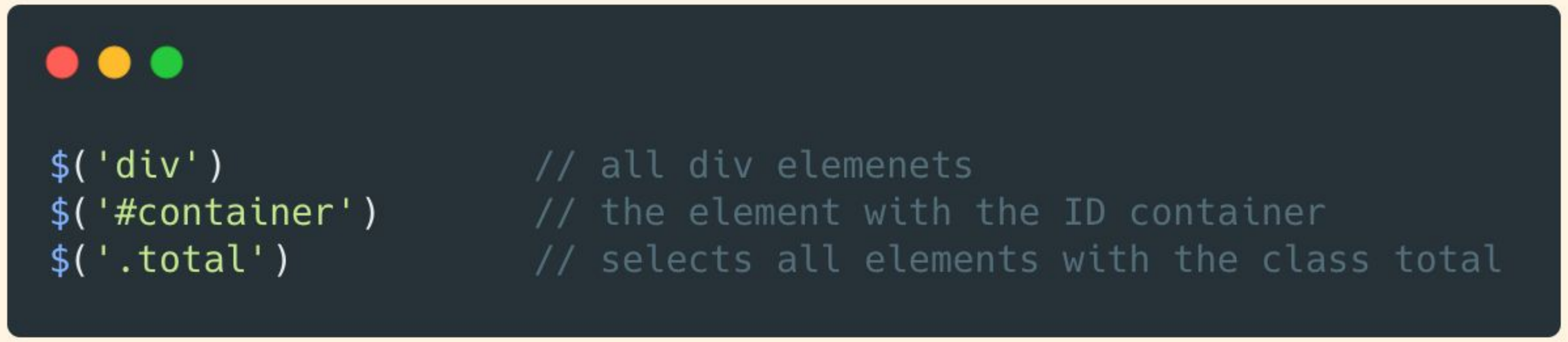
JQUERY

WHAT IS JQUERY?

jQuery is a widely used JavaScript library to help you find and change html elements on web pages, and do things in response to user events.

SELECTORS

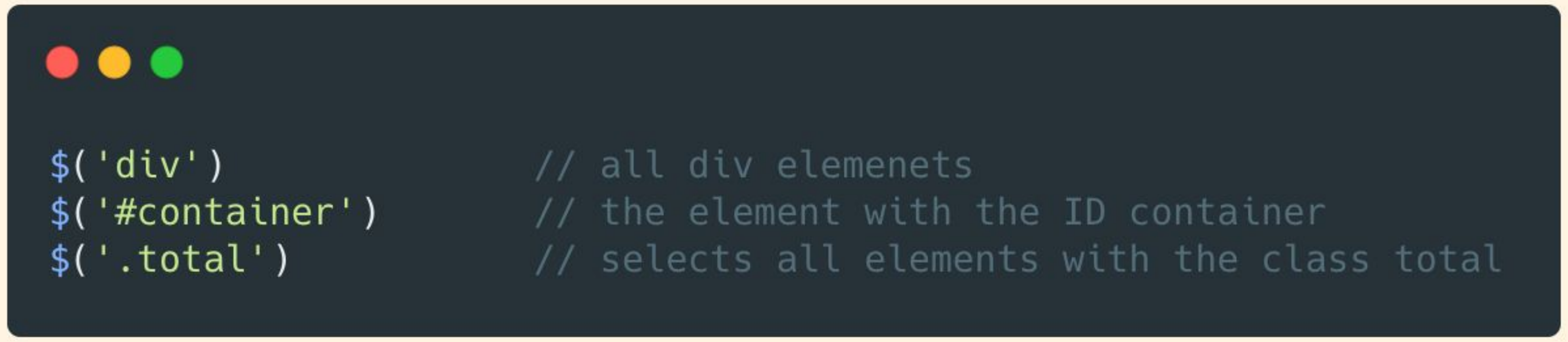
jQuery allows you to identify the HTML element that you want to work with using `CSS` selectors and uses the `$()` function to do this. We write a jQuery selector by passing the `$()` function a selector as a string, like this:



```
$('div')                // all div elemenets
$('#container')         // the element with the ID container
$('.total')             // selects all elements with the class total
```

JQUERY OBJECTS

The `$()` function returns a `jQuery object`. This object refers to the elements that you selected. You can then call jQuery methods on this object to inspect or change those elements.



```
$('div')              // all div elemenets
$('#container')       // the element with the ID container
$('.total')           // selects all elements with the class total
```

WE’RE BUILDING A TODO LIST 🗒️

BUILDING A TODO LIST

We'll be using jQuery and JavaScript functions to create a small todo list.

Let's start with making the 'Add to list' button work

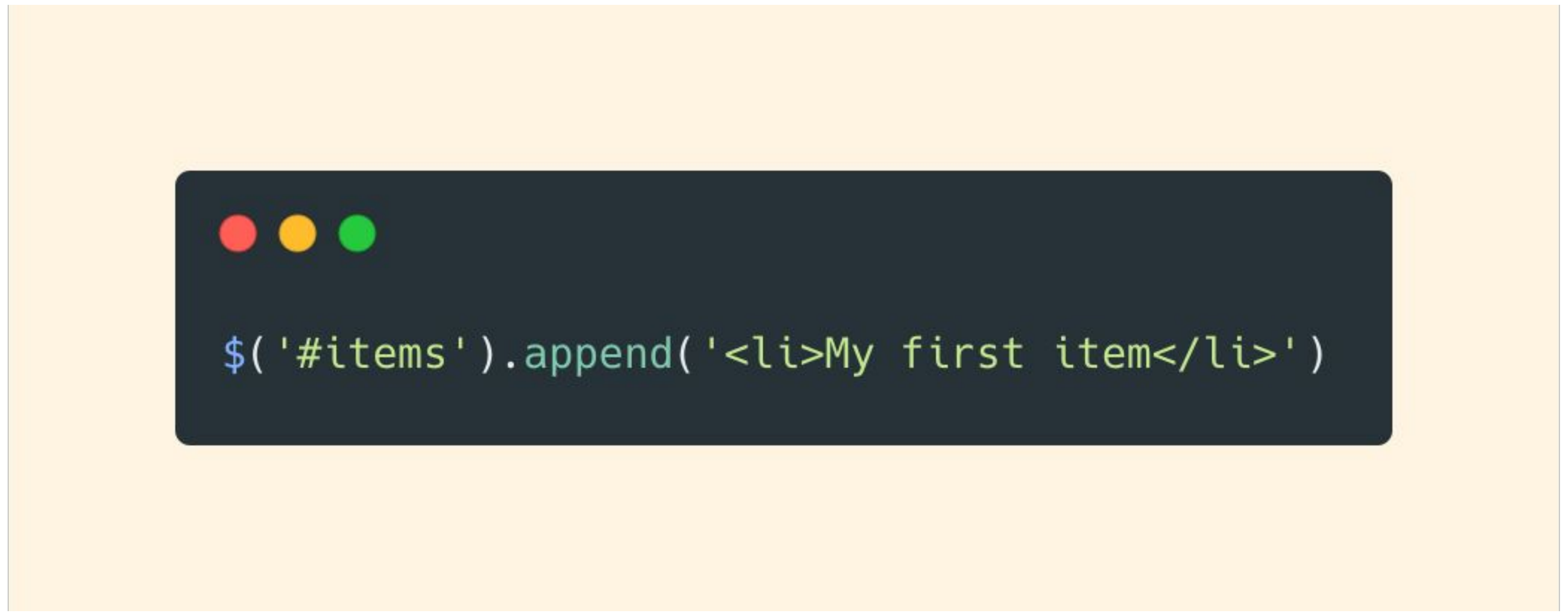
ADDING ITEMS TO THE LIST: CONSOLE

1. Open the `index.html` in Google Chrome
2. Open the browser’s console

There's actually a list on the page with the id `items`, but you can't see it yet because it's empty.

ADDING ITEMS TO THE LIST: CONSOLE

Let's add an item to the list by typing the following into the browser:



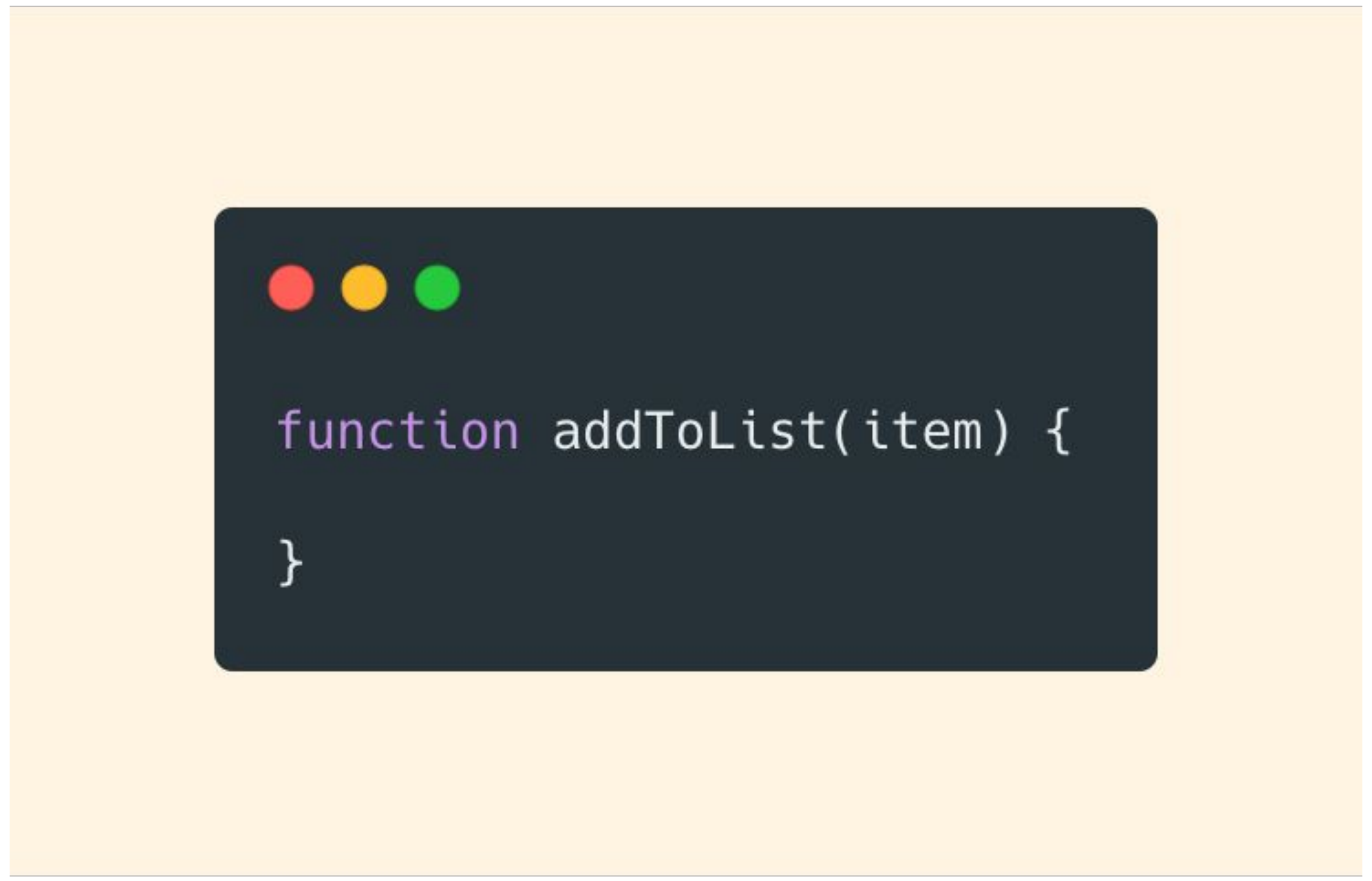
```
$('#items').append('<li>My first item</li>')
```

WHAT DID WE JUST DO?

The `$('#items')` function returned a jQuery object that has selected the html element with id `items`. You then called the `append` method on this object. The append method takes an html string as a parameter, and appends it to the element that you selected.

ADDING ITEMS TO THE LIST:
SCRIPT.JS

We're going to create a function to add items to the list.
Open the `script.js` and we'll begin the function with:

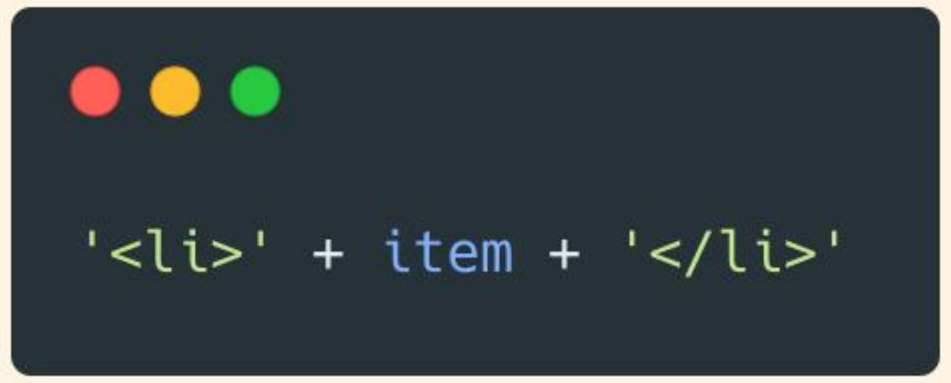


```
function addToList(item) {

}
```

ADDING ITEMS TO THE LIST:
SCRIPT.JS

The code you want to put in this function looks a lot like what you wrote on the console earlier, but instead of adding 'My first item', we want to add the parameter to this function:

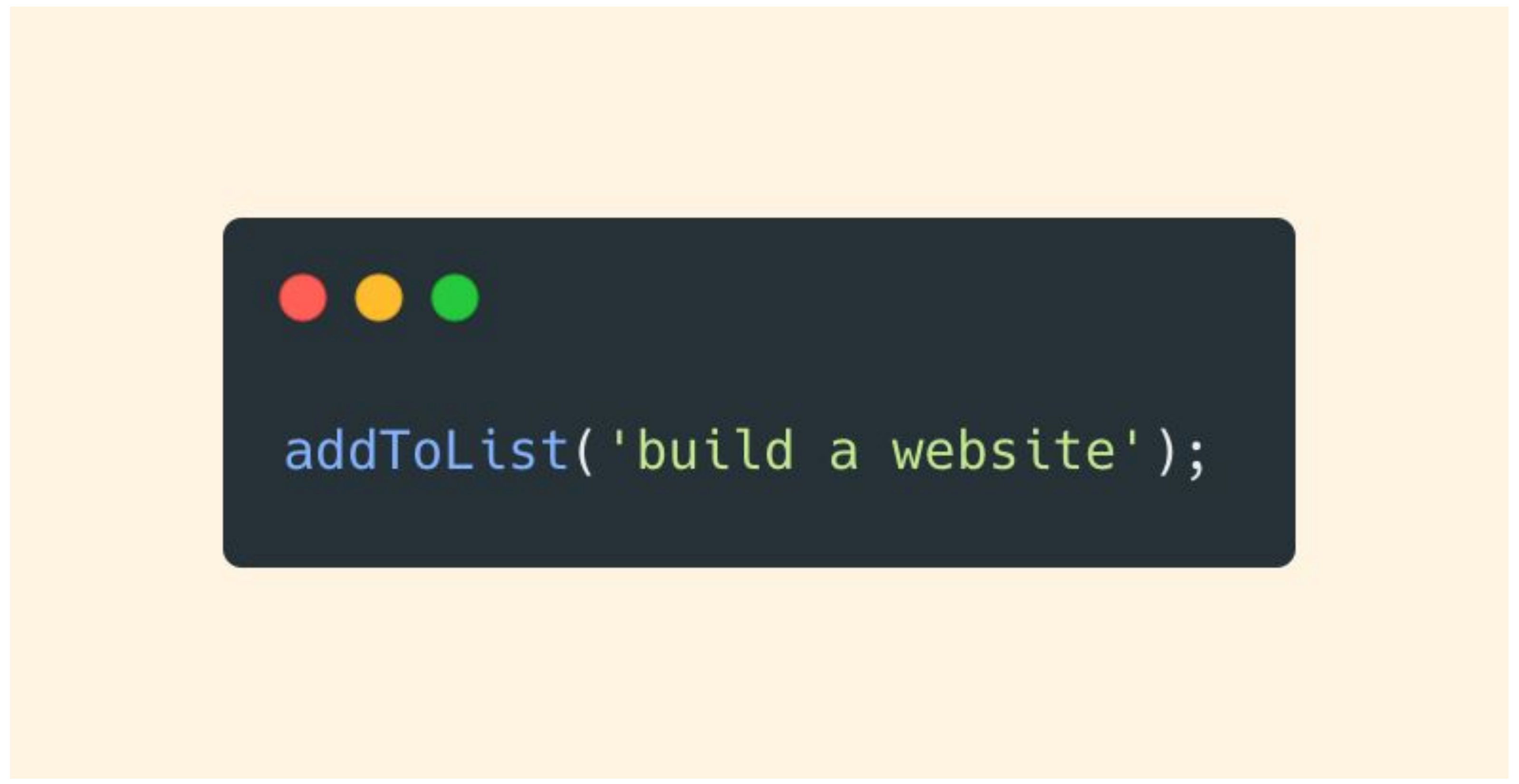


```
'<li>' + item + '</li>'
```

ADDING ITEMS TO THE LIST:
SCRIPT.JS

BLACK CODHER

Once you've done that, reload index.html in your browser, and try running the function you've just written from the browser console and you should see it added to the list.



```
addToList('build a website');
```

MAKING THE INPUT FIELD WORK:
CONSOLE

There's an input field and button on the page. We're now going to connect those to the function you just wrote.

1. Find the `input` element in the `index.html` and remember its id
2. With the console open in the browser, type some text in the input field.
3. In the console write a `$()` selector that selects the input (remember the id from step 1) and call the `.val()` method on that object.

You should now see the content that you typed into the input field

MAKING THE INPUT FIELD WORK:
CONSOLE

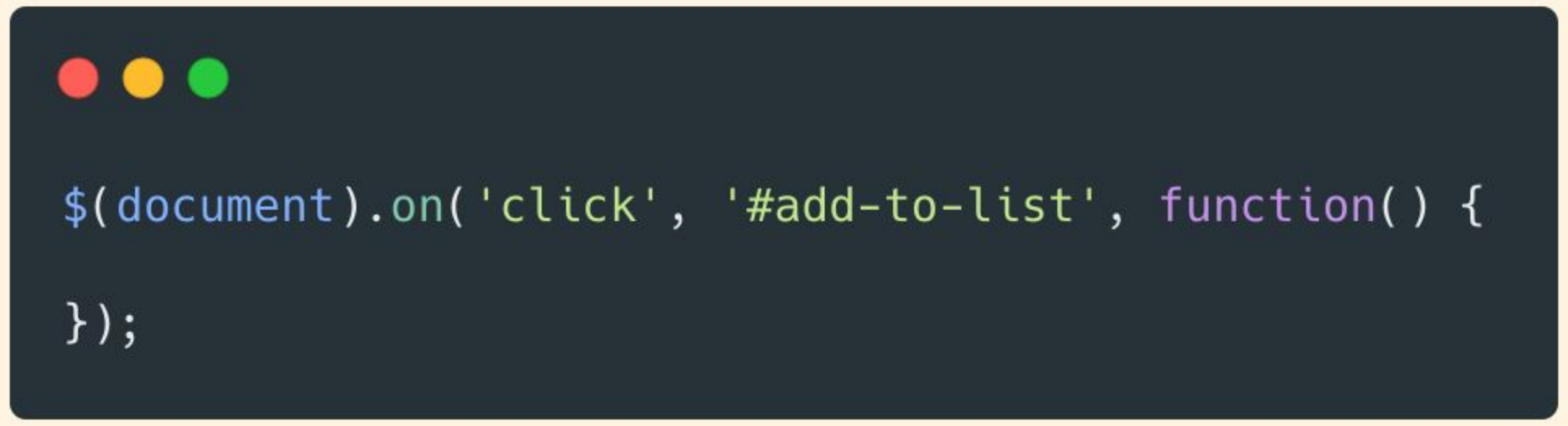
- Call the same method but with a string parameter and see what happens
- Add the `addToList` function to this and tie it all together:
  - Call the `addToList` with the `.val()` as a parameter

You should see that the text from the input field added to your todo list

MAKING THE INPUT FIELD WORK:
SCRIPT.JS

Now you know how to get the contents of the input box, and change it. Next you need to make all this happen when the button is clicked.

We can use the `click` method to add an `event handler` that will be called when the user clicks on something. Add this new function to `script.js`:



```
$(document).on('click', '#add-to-list', function() {
});
```

MAKING THE INPUT FIELD WORK:
`SCRIPT.JS`

BLACK CODHER

Inside this function, add the line that you worked out earlier which calls `addToList` with the current value of the input box.

That's all you need! Check that it works in your browser. You should be able to type something into the input box, click the button, and it will be added to the list.

When you've got that working, add another line to your `click` event handler, that empties the input box after adding the item to the list.

WHY JQUERY, WHY?

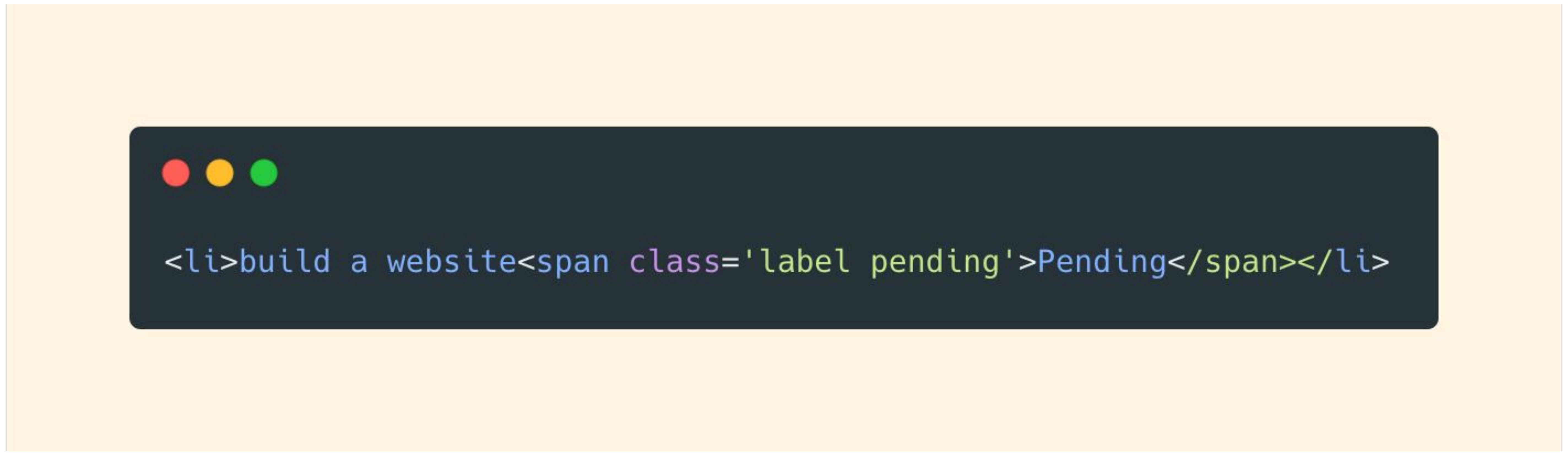
After you click on the button, the cursor is no longer in the input box. Use jQuery's `focus()` method to place the cursor back in the text field after clicking the button.

You can find code examples of how to use the `focus()` method in the jQuery documentation

CREATING LABELS

We're now going to add labels to the items in the list, so that you can mark them as done.

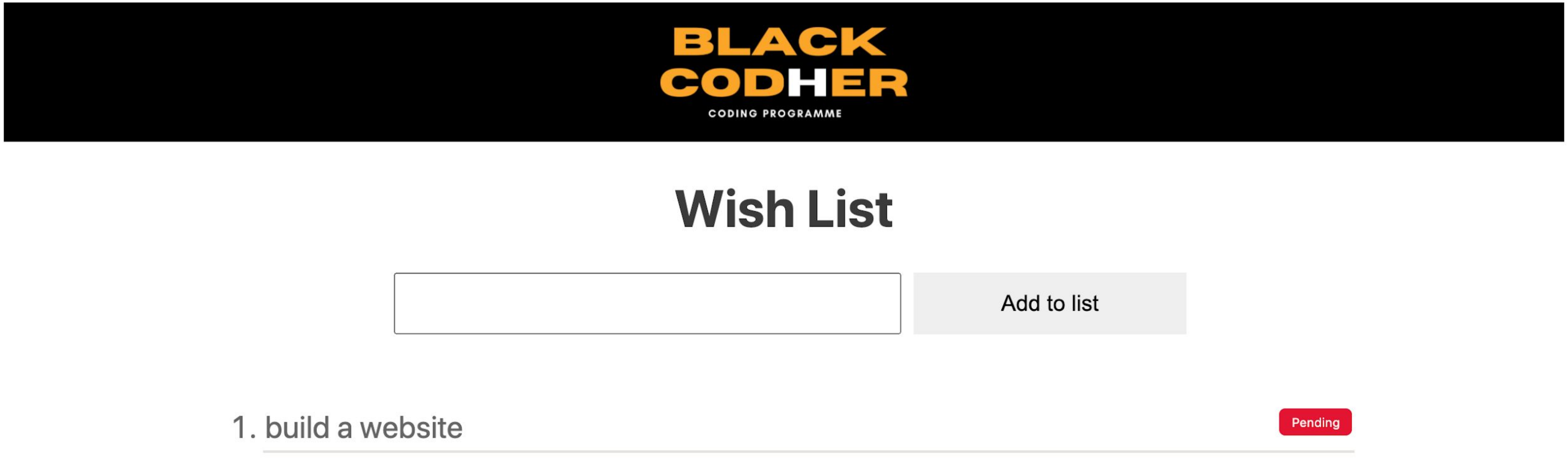
Start by adding a 'pending' label to each item when it gets added to the list. Change your `addToList` function so that when you call `addToList('build a website')`, it adds this html code:



```html
<li>build a website<span class='label pending'>Pending</span></li>
```

CREATING LABELS

Refresh the page and try it again. You should see something that looks like this:



MARKING ITEMS AS COMPLETED

When we click on the 'Pending' label, we want to mark items as complete. We shall do this by removing the 'Pending' label and adding a new 'Done' label.

Start by making a new click event handler for the '`.pending`' class.

The code we put inside the event handler this time is a little more interesting, because we want to change the item that was clicked on.

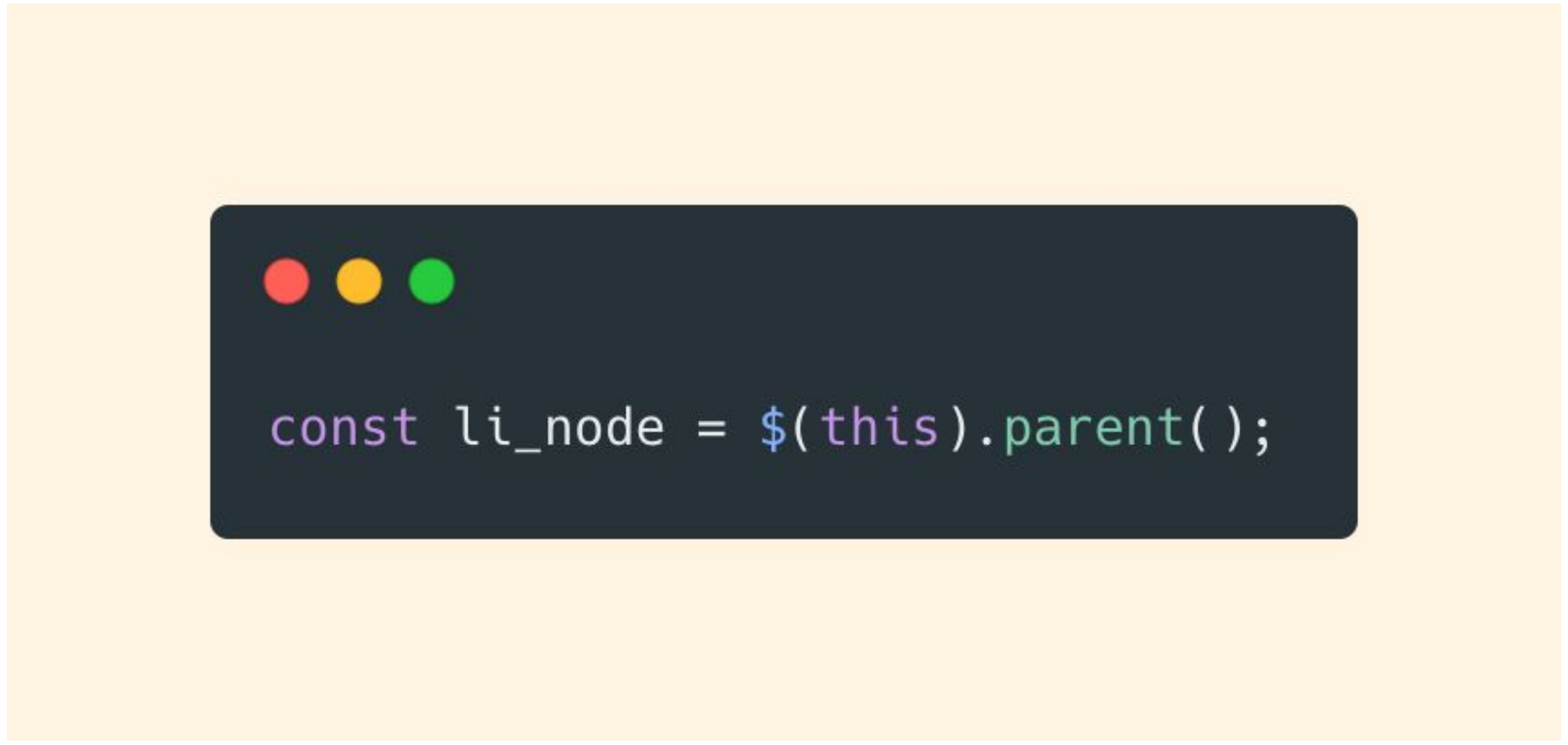
WHAT IS `this`?

JavaScript gives you the element that the event came from in a special variable called `this`. Because `this` comes from JavaScript itself, it is not a jQuery object. We can fix that by passing it to the `$()` function, as `$(this)`.

In our case, the element that the event came from is the `<span class='label pending'>` element that was clicked on. We can use the jQuery `.parent()` method to find the parent of that element, which will be the `<li>`.

MARKING ITEMS AS COMPLETED

So, begin your new event handler with this line:



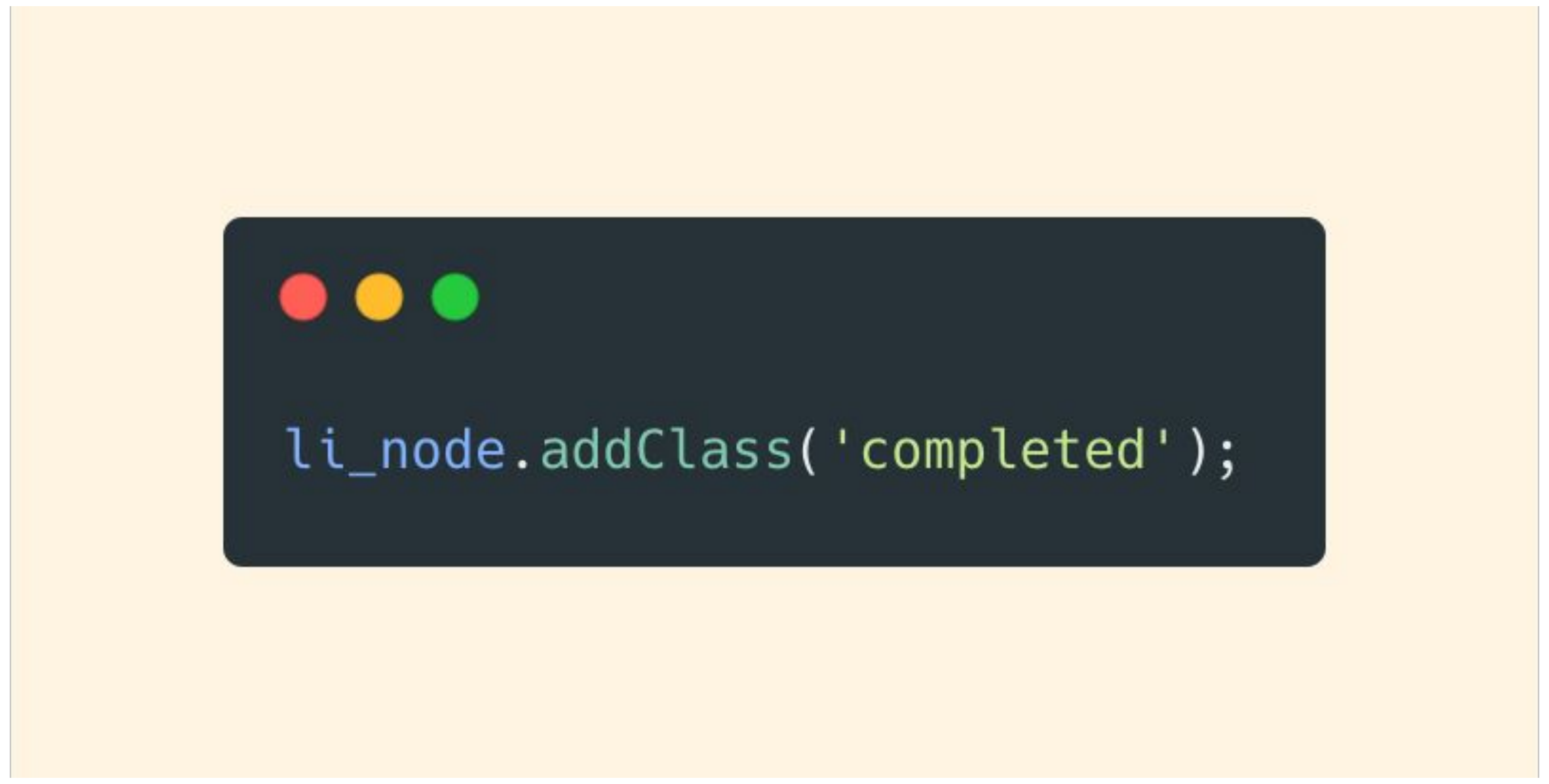
```
const li_node = $(this).parent();
```

MARKING ITEMS AS COMPLETED

Now that you have the right list item, use `.append()` to add a new label `<span class='label success'>Done!</span>`, and then use `.remove()` to remove the Pending label.

LET’S MAKE IMPROVEMENTS

We can make this look a little better. A useful technique is to add and remove css classes. Try adding this line to your event handler:



```
li_node.addClass('completed');
```

LET’S MAKE IMPROVEMENTS

In your web browser, use the 'Inspect element' feature to look at a list item, then click on the Pending label.

See how it now has `class="completed"` in the inspector, and is now getting styled by things in the css which apply to that class?

This approach lets us keep all our styles in css files, and have javascript turn them on and off.

SHOWING THE TOTAL TASK - CONSOLE

1. In the browser, add some items to the list
2. Open the console
3. Write a jQuery selector that selects all the pending labels. You can use the `.length` property to find out how many elements on the page a jQuery object refers to.
4. Store those lengths in two variables called `pending` and `completed`

Now it's time to display them on the page:

1. Look at `index.html` and find the `span` that comes just above the `<ol>` element.
2. Write a jQuery selector for that element, and then call `.text('Pending: ' + pending + ' Completed: ' + completed)` on it.

SHOWING THE TOTAL TASK - `SCRIPT.JS`

You now know how to count the number of items in the list and display the totals. Write a new function called `updateTotal` to do this.

Update the displayed totals by calling `updateTotal()`.

1. After adding an item to the list
2. When changing the state of an item from **Pending** to **Done**

SUMMARY

1. The `$()` function creates a jQuery object that you can call methods on. You can pass it a selector, like `$('#add-to-list')` or `$('.pending')`, or a javascript object like `$(this)` or `$(document)`.
2. The jQuery methods `.append()` and `.remove()` can be used to add and delete elements on the page.
3. jQuery methods like `.text()`, `.val()`, and `.addClass()` can be used to get things from the page and change them.
4. The `.on()` function can be used to add an event handler. By making a handler for the click event, you can run your code when somebody clicks on something on the page.